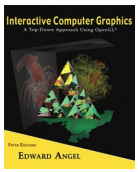


Name: _____ **Term # ____**

Homework 8 SOLUTIONS

(400 points)

NOTE: **Chapter 8 of the textbook shows the texture mapping, antialising, composition and alpha blending techniques made possible by using discrete buffers..**
Part A is intended to be done by hand.
Part B is an OpenGL application.

A. (300 pts) Paper and Pencil

(Guidelines: *Read the material from the textbook chapter, you can use textbook figures to exemplify your answer, use keywords, summarize your answer, but the answer* ***cannot be longer the 7 lines!****)*

8.1 BUFFERS
a. Explain.
ANSWER:
(Angel pp. 383)

8.2 DIGITAL IMAGES
a. Explain.
ANSWER:
(Angel pp. 385)

8.3 WRITING IN BUFFERS
a. Explain.
ANSWER:
(Angel pp. 388)

8.3.1 Writing Modes
a. Explain.
ANSWER:
(Angel pp. 389)

8.3.2 Writes with XOR
a. Explain writing with XOR and Figure 8.7
ANSWER:
(Angel pp. 391)

8.4 BIT AND PIXEL OPERATIONS IN OPENGL
a. Explain.
ANSWER:
(Angel pp. 392)

8.4.1 OPENGL Buffers and the Pixel Pipeline
a. Explain.
ANSWER:
(Angel pp. 392)

8.4.2 Bitmaps
a. Explain.
ANSWER:
(Angel pp. 394)

8.4.3 Raster Fonts
a. Explain.
ANSWER:
(Angel pp. 395)

8.4.4 Pixels and Images
a. Explain.
ANSWER:
(Angel pp. 396)

8.4.5 Lookup Tables
a. Explain lookup tables and Figure 8.11.
ANSWER:
(Angel pp. 398)

8.6 MAPPING METHODS
a. Explain.
ANSWER:
(Angel pp. 401)

8.7 TEXTURE MAPPING
a. Explain.
ANSWER:
(Angel pp. 401)

8.8 TEXTURE MAPPING IN OPENGL
a. Explain.
ANSWER:
(Angel pp. 410)

8.8.4 Texture Objects
a. Explain.
ANSWER:
(Angel pp. 410)

8.9 TEXTURE GENERATION
a. Explain.
ANSWER:
(Angel pp. 421)

8.10 ENVIRONMENT MAPS
a. Explain.
ANSWER:
(Angel pp. 422)

8.11 COMPOSITING TECHNIQUES
a. Explain.
ANSWER:
(Angel pp. 427)

8.11.1 Opacity and Blending
a. Explain.
ANSWER:
(Angel pp. 428)

8.11.2 Image Compositing
a. Explain.
ANSWER:
(Angel pp. 427)

8.11.3 Blending and Compositing in OpenGL
a. Explain.
ANSWER:
(Angel pp. 429)

8.11.4 Antialising Revisited
a. Explain.
ANSWER:
(Angel pp. 431)

8.11.5 Back-to-front and Front-to-Back Rendering
a. Explain
ANSWER:
(Angel pp. 432)

8.11.6 Depth Cueing and Fog
a. Explain.
ANSWER:
(Angel pp. 433)

8.12 MULTIRENDERING AND THE ACCUMULATION BUFFER
a. Explain.
ANSWER:
(Angel pp. 434)

8.12.1 Scene Antialising
a. Explain.
ANSWER:
(Angel pp. 435)

8.12.2 Bump Mapping and Embossing
a. Explain.
ANSWER:
(Angel pp. 436)

8.12.3 Image Processing
a. Explain.
ANSWER:
(Angel pp. 436)

8.13 SAMPLING AND ALIASING
a. Explain.
ANSWER:
(Angel pp. 439)

B. (100 pts) Visual Studio 2008 C++ Project

B1. Create Visual Studio 2008 C++, Empty Project, Homework8:

```
//TexturedCube.c

Apply your favorite texture to any of the cubes we have created in the
previous Homeworks.
```

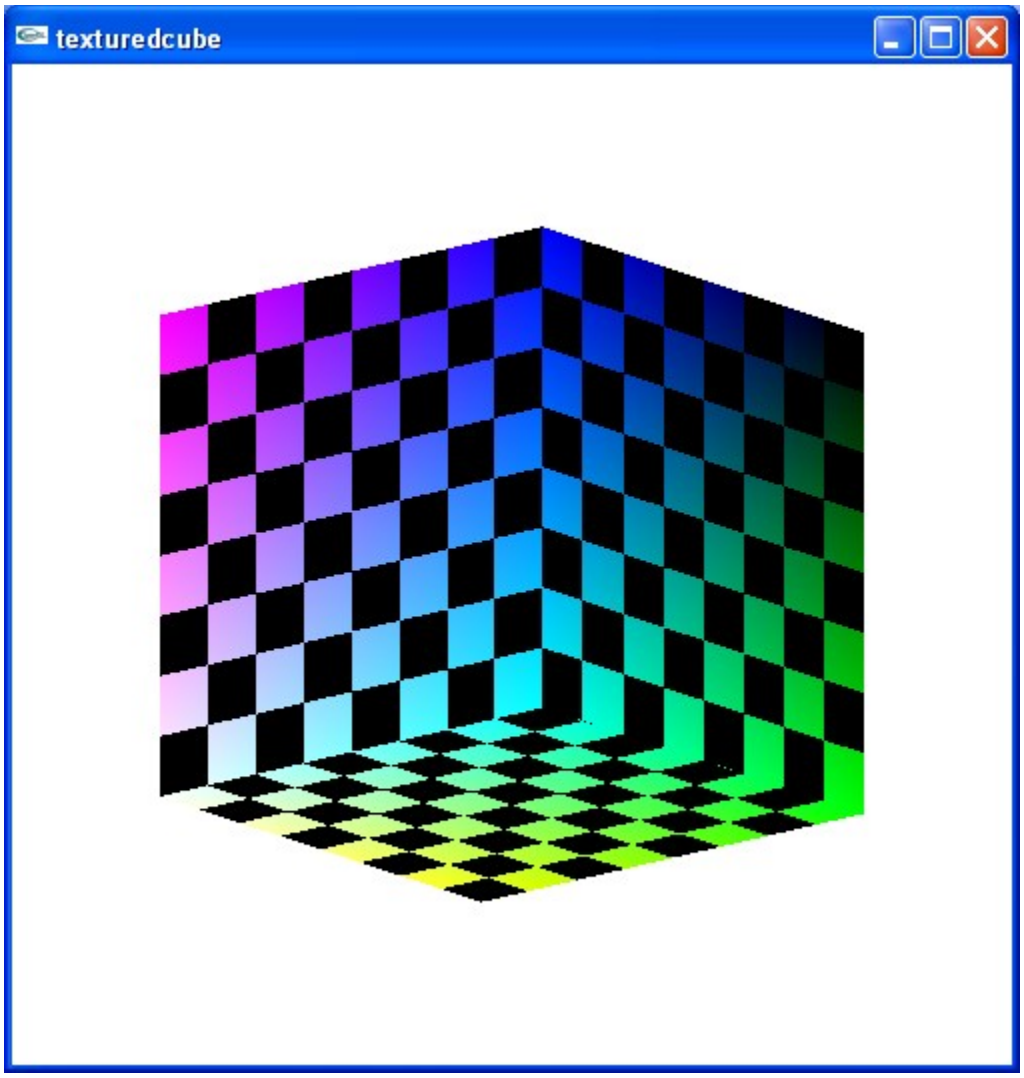
Build and run this Project: Insert a screenshot of your output.

ANSWER:

```
/ TexturedCube.c

#include <stdlib.h>

#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif

GLfloat planes[]= {-1.0, 0.0, 1.0, 0.0};
GLfloat planet[]= {0.0, -1.0,  0.0, 1.0};
GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},
    {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},
    {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};
GLfloat colors[][4] = {{0.0,0.0,0.0,0.5},{1.0,0.0,0.0,0.5},
    {1.0,1.0,0.0,0.5}, {0.0,1.0,0.0,0.5}, {0.0,0.0,1.0,0.5},
    {1.0,0.0,1.0,0.5}, {1.0,1.0,1.0,0.5}, {0.0,1.0,1.0,0.5}};

void polygon(int a, int b, int c, int d)
{
    glBegin(GL_POLYGON);
    glColor4fv(colors[a]);
    glTexCoord2f(0.0,0.0);
    glVertex3fv(vertices[a]);
    glColor4fv(colors[b]);
    glTexCoord2f(0.0,1.0);
    glVertex3fv(vertices[b]);
    glColor4fv(colors[c]);
    glTexCoord2f(1.0,1.0);
    glVertex3fv(vertices[c]);
    glColor4fv(colors[d]);
    glTexCoord2f(1.0,0.0);
    glVertex3fv(vertices[d]);
    glEnd();
}

void colorcube()
{

/* map vertices to faces */

    polygon(0,3,2,1);
```

```
    polygon(2,3,7,6);
    polygon(0,4,7,3);
    polygon(1,2,6,5);
    polygon(4,5,6,7);
    polygon(0,1,5,4);
}

static GLfloat theta[] = {0.0,0.0,0.0};
static GLint axis = 2;

void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glRotatef(theta[0], 1.0, 0.0, 0.0);
    glRotatef(theta[1], 0.0, 1.0, 0.0);
    glRotatef(theta[2], 0.0, 0.0, 1.0);
    colorcube();
    glutSwapBuffers();
}

void spinCube()
{
    theta[axis] += 2.0;
    if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
    glutPostRedisplay();
}

void mouse(int btn, int state, int x, int y)
{
    if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN) axis = 0;
    if(btn==GLUT_MIDDLE_BUTTON && state == GLUT_DOWN) axis = 1;
    if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN) axis = 2;
}

void myReshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho(-2.0, 2.0, -2.0 * (GLfloat) h / (GLfloat) w,
            2.0 * (GLfloat) h / (GLfloat) w, -10.0, 10.0);
    else
        glOrtho(-2.0 * (GLfloat) w / (GLfloat) h,
            2.0 * (GLfloat) w / (GLfloat) h, -2.0, 2.0, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
}

void key(unsigned char k, int x, int y)
{
    if(k == '1') glutIdleFunc(spinCube);
    if(k == '2') glutIdleFunc(NULL);
    if(k == 'q') exit(0);
}

int main(int argc, char **argv)
```

```
{
   GLubyte image[64][64][3];
   int i, j,  c;
   for(i=0;i<64;i++)
   {
     for(j=0;j<64;j++)
     {
       c = ((((i&0x8)==0)^((j&0x8))==0))*255;
       image[i][j][0]= (GLubyte) c;
       image[i][j][1]= (GLubyte) c;
       image[i][j][2]= (GLubyte) c;
     }
   }
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("TexturedCube");
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutIdleFunc(spinCube);
    glutMouseFunc(mouse);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_TEXTURE_2D);
    glTexImage2D(GL_TEXTURE_2D,0,3,64,64,0,GL_RGB,GL_UNSIGNED_BYTE,
image);
    glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT);
    glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT);
    glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
    glTexParameterf(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
    glutKeyboardFunc(key);
    glClearColor(1.0,1.0,1.0,1.0);
    glutMainLoop();
}
```


texturedcube